Part II – [3 marks] Design and Implementation of an Interrupts Simulator

The objective of this section of the lab was to assess the impact of different interrupt handling parameters on total execution time within a simulated operating system environment.

The simulation incorporated all steps of the interrupt sequence. This included: entering kernel mode, context switching, locating and loading the ISR (interrupt service routine) addresses, executing the ISR, and the transitions back to user mode.

There were twenty automated test runs conducted with varying parameters. These included the context save and restore times of 10, 20, and 30 ms, and the ISR execution times of 40 to 200 ms. For the additional two test runs, the parameters included 4 byte vector addresses and a high speed processor. Each test run performance was tracked into log execution records for every step.

As the context save and restore time extended from 10 ms to 30 ms, the total program time increased in an almost proportional manner. This was due to the additional time caused by every interrupt call proving the scope of context management and system overhead.

As the time parameters for the ISR increased, the total execution time increased in a rapid, and non-linear, manner. Long ISR times caused an increase in user processes suspension time which resulted in increased total simulated time. Above 160 ms, the ISR time as a whole dominated the elapsed time proving inefficient circuit drivers can bottleneck the system.

The 4-byte vector-address situation added a little more lookup and memory-access overhead. Although each lookup took just a couple of milliseconds, the effect during times of heavy interrupt loads became more pronounced.

The reduced size of context and ISR delays became a function of having a faster CPU. With faster execution cycles, the same interrupt processes took a smaller proportion of the total runtime. This suggests that CPU speed offers improvements in user-mode efficiency more than it does in kernel-mode latency.

System performance is fundamentally limited by the execution time of ISRs and the delays caused by context switches. While a bigger vector-address size will incur very minimal overhead, a quicker CPU will lower delays, but will not eliminate kernel delays. Good system activity relies on well-designed ISRs and better context switching.

Repository: https://github.com/md27-cpu/SYSC4001_A1.git